Docket No.: M062
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Michael J. Sundermeyer et al.

Application No.: 10/690,214 Confirmation No.: 2346

Filed: October 21, 2003 Art Unit: 2176

For: WEB SITE MANAGEMENT LIFECYCLE Examiner: M. K. Botts

APPEAL BRIEF

MS Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

As required under 37 C.F.R. § 41.37(a), this Appeal Brief is being submitted within two months of the Notice of Appeal filed December 5, 2006, and is in furtherance of said Notice of Appeal. The fees required under 37 C.F.R. § 41.20(b)(2) are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF. This Appeal Brief contains items under the following headings, as required by 37 C.F.R. § 41.37 and M.P.E.P. § 1205.02:

I. REAL PARTY IN INTEREST

The real party in interest for this appeal is:

Adobe Systems Incorporated.

II. RELATED APPEALS AND INTERFERENCES

There are no other appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

A. Total Number of Claims in Application

There are 31 claims pending in application.

B. Current Status of Claims

1. Claims canceled: None

2. Claims withdrawn from consideration but not canceled: None

3. Claims pending: 1-31

4. Claims allowed: None

5. Claims rejected: 1-31

C. Claims On Appeal

The claims on appeal are claims 1-31

IV. STATUS OF AMENDMENTS

A Final Office Action (hereinafter *Final Office Action*) rejecting the claims of the present application was mailed on September 8, 2006. In response, Appellant did not file an Amendment, but instead filed a Notice of Appeal, which this Appeal Brief supports. Accordingly, the claims on appeal are the same as those rejected in the *Final Office Action*. A listing of the claims on appeal is provided under the "Claims Appendix" heading of this Appeal Brief.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The following provides a concise explanation of the subject matter defined in the independent claim involved in the appeal, referring to the Specification by page and line number and to the Drawings by reference characters, as required by 37 C.F.R. § 41.37. As such, each element of the claims is identified by a reference to the Specification and Drawings, where applicable. However, citation to passages in the Specification and Drawings does not imply that elements recited therein should be read into corresponding claim elements.

According to one claimed embodiment, such as that of independent claim 1, a method for maintaining a Web site comprises browsing to a Web page to be edited (*See e.g.*, Specification at ¶ [0029]; Fig. 3, step 300), automatically downloading a source file for said Web page from a file transfer server related to said Web page including related files associated with display of said Web page (*See e.g.*, Specification at ¶ [0030]; Fig. 3, step 303), editing said source file (*See e.g.*, Specification at ¶ [0030]; Fig. 3, step 304), and automatically publishing said edited source file to said file transfer server associated with said Web site including said related files associated with said display of said Web page (*See e.g.*, Specification at ¶ [0030]; Fig. 3, step 307).

According to another claimed embodiment, such as that of independent claim 11, a computer program product having a computer readable medium with computer program logic recorded thereon for managing a Web site (*See e.g.*, Specification at ¶ [0036]; Fig. 5, item 500) comprises code for browsing to a Web page to be edited (*See e.g.*, Specification at ¶ [0029]; Fig. 3, step 300), code for automatically retrieving from a file transfer server one or more Web source files for at least one of said Web page and one or more Web page-dependent files (*See e.g.*, Specification at ¶ [0030]; Fig. 3, step 303), code for editing said one or more Web source files (*See e.g.*, Specification at ¶ [0030]; Fig. 3, step 304), and code for automatically uploading said edited one or more Web source files to said file transfer server (*See e.g.*, Specification at ¶ [0030]; Fig. 3, step 307).

According to another claimed embodiment, such as that of independent claim 20, a method for managing a Web site lifecycle from a graphical user interface (GUI) comprises displaying a Web browser in a first window of said GUI (*See e.g.*, Specification at ¶ [0023];

Fig. 1A, item 100), wherein a user browses on said Web browser to locate a Web page to be edited (*See e.g.*, Specification at ¶ [0023]), selecting an edit indicator displayed on said GUI indicating a preference to edit said Web page (*See e.g.*, Specification at ¶ [0024]; Fig. 1B, item 103), responsive to said selecting, transitioning said first window to display a page editor (*See e.g.*, Specification at ¶ [0025]; Fig. 1C, item 105), choosing a publish indicator displayed on said GUI indicating to publish said edited Web page (*See e.g.*, Specification at ¶ [0025]; Fig. 1C, item 107), and responsive to said selecting, transitioning said first window back to display said Web browser (*See e.g.*, Specification at ¶ [0027]; Fig. 1D, item 100).

According to yet another claimed embodiment, such as that of independent claim 21, a Web page editor comprises a graphical interface (GI) for receiving interaction from a user (*See e.g.*, Specification at ¶ [0023]; Fig. 1A, item 10), a Web browser displayed to said user in a main window of said GI (*See e.g.*, Specification at ¶ [0023]; Fig. 1A, item 100), an edit indicator associated with said display of said Web browser and presented to said user on said GI (*See e.g.*, Specification at ¶ [0024]; Fig. 1B, item 103), wherein said edit indicator controls execution of retrieval logic (*See e.g.*, Specification at ¶ [0024]), an edit screen replacing said Web browser in said main window responsive to said user selecting said edit indicator (*See e.g.*, Specification at ¶ [0025]; Fig. 1C, item 105), wherein said user makes edits to said Web page (*See e.g.*, Specification at ¶ [0025]), and a publish indicator associated with a display of said edit screen and presented to said user on said GI, wherein said publish indicator controls execution of said upload logic (*See e.g.*, Specification at ¶ [0025]; Fig. 1C, item 107).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. First Grounds

Claims 1-19 are rejected under 35 U.S.C. § 102(b) as being anticipated by GlobalScape, "CuteFTP Pro Technical Overview," White Paper, May 22, 2001 (hereinafter *CuteFTP*).

B. Second Grounds

Claim 20 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Altova Inc. & Altova GmbH, "XML Spy 4.0 Manual," September 10, 2001 (hereinafter *XML Spy*).

C. Third Grounds

Claims 21-31 are rejected under 35 U.S.C. § 103(a) as being unpatentable over *XML Spy* in view of *CuteFTP*.

VII. ARGUMENT

Appellant respectfully traverses the outstanding rejections of the pending claims, and requests that the Board reverse these rejections in light of the remarks contained herein. The claims do not stand or fall together, and Appellant presents separate arguments for several claims. Each of the separately argued claims are presented with separate headings and sub-headings in accordance with 37 C.F.R. § 41.37(c)(1)(vii).

A. First Grounds (Claims 1-19)

Claims 1-19 are rejected under 35 U.S.C. § 102(b) as being anticipated by *CuteFTP*. In order to anticipate a claim under 35 U.S.C. § 102, a single reference must teach each and every element of the claim. *See Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631 (Fed. Cir. 1987). Appellant respectfully submits that *CuteFTP* fails to teach each and every element of claims 1-19, and respectfully requests that the Board reverse these rejections.

1. Independent Claims 1 and 11

Independent claims 1 and 11 require, in part, "browsing to a Web page to be edited . . . ." The Examiner did not addressed this element in his rejections. *See e.g., Final Office Action*, pp. 3 and 19. Appellant points out that *CuteFTP* only discloses an FTP client, and that merely transferring a file to or from a server via an FTP client, even if that file is an HTML file, is not the same as "browsing to a Web page to be edited." *See CuteFTP* at p. 12. Therefore, *CuteFTP* does not teach or suggest each and every element of claims 1 and 11.

Independent claims 1 and 11 also require, in part, "automatically downloading a source file for said Web page from a file transfer server related to said Web page including related files associated with display of said Web page." *CuteFTP* discloses automatically downloading or uploading log files. *CuteFTP* at p. 7. However, *CuteFTP*'s log file is not "a source file for a Web page [that has been browsed to] . . . ." *See id.* Thus, *CuteFTP* does not teach or suggest "automatically downloading a source file for said Web page [that has been browsed to] . . . ," much less "automatically downloading related files associated with display of said Web page," as required by claims 1 and 11. Appellant also points out that the Examiner's attempt to rebut these arguments has been non-responsive. *See Final Office*

Action, p. 19. Particularly, the Examiner states that "[n]either claim 1 nor claim 11 specifies that 'a log file is the source file for a web page that has been browsed to.'" *Id.* However, Appellant has never stated that claims 1 and 11 require a log file. Instead, Appellant has repeatedly asserted that *CuteFTP*'s log file does not meet the claimed source file, at least, because *CutFTP*'s log file is not a source file for a web page that was been browsed to.

Independent claims 1 and 11 further require, in part, "automatically publishing said edited source file to said file transfer server associated with said Web site including said related files associated with said display of said Web page." The Examiner relies upon *CuteFTP* as teaching or suggesting the periodic updating of a web site. *Final Office Action* at p. 3. However, as noted above, *CuteFTP* does not teach or suggest that the file to be periodically uploaded is a source file for a Web page that has been browsed to. In addition, *CuteFTP* does not teach or suggest automatically publishing related files associated with display of the Web page, as required by claims 1 and 11. Again, the Examiner's attempt to rebut these arguments has been non-responsive. *See Final Office Action* at p. 19. Appellant has not stated that claims 1 and 11 require that a file be periodically uploaded. Instead, Appellant has asserted that *CuteFTP*'s periodic uploading does not meet the claimed publishing because *CuteFTP*'s file (to be periodically uploaded) is neither a source file nor a file associated with display of a Web page browsed to.

Dependent claims 2-10 and 12-19 depend, either directly or indirectly from claims 1 or 11, respectively, thus inheriting all the elements of their respective independent claims. As noted above, *CuteFTP* does not teach every element of independent claims 1 and 11. Thus, *CuteFTP* also fails to teach each and every element of dependent claims 2-10 and 12-19. Moreover, each of these dependent claims recite additional elements not taught or suggested by the applied art, as separately argued under the following sub-headings.

2. Dependent Claims 2 and 12

Dependent claim 2 recites "scanning said Web page for page-dependent related files prior to said automatically downloading." Claim 12 recites similar elements. The passage of *CuteFTP* relied upon by the Examiner as meeting this element discloses a combination of "scheduler and directory mirroring, [by which] users can accomplish recurring or continuous mirroring of a particular directory." *CuteFTP* at p. 9, *cited in Final Office Action* at p. 4.

Appellant respectfully asserts that merely performing directory mirroring of an entire folder is very different from scanning a Web page for page dependent files. For instance, related files need not be in the same folder as the Web page itself. Meanwhile, unrelated files may be located in the same folder as the Web page. Therefore, merely mirroring a directory is not the same, or even similar, to the elements recited in claims 2 and 12.

3. Dependent Claims 3 and 13

Dependent claim 3 recites "scanning said edited source file for modified page-dependent related files prior to said automatically publishing." Claim 13 recites similar elements. The passage of *CuteFTP* relied upon by the Examiner as meeting this element discloses a combination of "scheduler and directory mirroring, [by which] users can accomplish recurring or continuous mirroring of a particular directory." *CuteFTP* at p. 9, *cited in Final Office Action* at p. 4. Appellant respectfully asserts that merely performing directory mirroring of an entire folder is very different from scanning a Web page for page dependent files. For instance, related files need not be in the same folder as the Web page itself. Meanwhile, unrelated files may be in fact located in the same folder as the Web page. Therefore, merely mirroring a directory is not the same, or even similar, to the elements recited in claims 3 and 13.

4. Dependent Claims 5 and 14

Dependent claim 5 recites "translating local links to said added page-related files to reflect a location of said added page-related files on said file server." Claim 14 recites similar elements. The Examiner states that *CuteFTP* discloses scanning a Web page for the files to be updated. *Final Office Action* at p. 5. However, Appellant has been unable to identify any specific passage of *CuteFTP* that discloses this element, and the Examiner has not shown otherwise. Appellant respectfully asserts that *CuteFTP* only discloses mirroring a directory, and that it does not translate local links to added page-related files to reflect a location of those files on a file server, as recited in claims 5 and 14.

5. Dependent Claims 6 and 15

Dependent claim 6 recites "mapping addresses of said edited source file and said related files associated with said display of said Web page to an address location

commensurate with said file transfer server." Claim 15 recites similar elements. The Examiner states that *CuteFTP* teaches mirroring, "which makes the content of the remote drive exactly like the contents of the local drive, vice versa, or both." *Final Office Action* at p. 5. However, merely mirroring an entire folder is not the same as mapping addresses of an edited source file and related files associated with the display of a Web page to an address location commensurate with a file transfer server, as recited in claims 6 and 15.

6. Dependent Claims 10 and 19

Dependent claim 10 recites "stripping said modifications to said one or more elements from said edited source responsive to said checking; and updating said database with said modifications." Claim 19 recites similar elements. The Examiner does not properly address these claims, and in fact rejects claims 10 and 19 on the same basis as the rejection of claim 5, even though claim 10 (and 19) recites entirely different elements from claim 5. Appellant respectfully asserts that *CuteFTP* does not disclose stripping modifications to one or more elements from an edited source responsive to a checking step and updating a database with the modifications, as recited in claims 10 and 19.

B. Second Grounds (Claim 20)

Claim 20 is rejected under 35 U.S.C. § 103(a) as being unpatentable over *XML Spy*. To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine references' teachings. Second, there must be a reasonable expectation of success. Finally, the prior art references must teach or suggest all the claim elements. M.P.E.P. § 2143. Without conceding the second criterion, Appellant asserts that the Examiner's rejection of these claims does not satisfy the first and third criteria.

1. Lack of Motivation

The rejection of claim 20 must be reversed because there is insufficient motivation to modify *XML Spy*. The Examiner states that:

> [i]t would have been obvious to one of ordinary skill in the art at the time of the invention to have used the function of XML Spy to monitor documents on a server, and to edit documents within a browser for a browser, to have used a browser to locate a Web page to be edited, for the obvious reason and beneficial purpose that XML Spy is obviously designed to cooperate with and use browsers in accessing and editing documents located on the Web.

Final Office Action at p. 12. Even assuming, *arguendo*, that *XML Spy* is "designed to cooperate with and use browsers," the Examiner has not shown why it would be desirable to modify *XML Spy*. Appellant respectfully points out that the Examiner's stated motivation is only a statement that the *XML Spy* ***can be*** modified. However, the mere fact that a reference ***can be*** modified does not render the resultant modification obvious unless the prior art also suggests the desirability of the modification. *See* M.P.E.P. § 2143.01. There is no suggestion or motivation, either in the prior art or in the knowledge available to a person of ordinary skill in the art, to modify *XML Spy*.

2. Lack of All Claimed Elements

Appellant respectfully points out that the Examiner has not specifically addressed many of the elements required by claim 20, which does not conform with Office policy. *See* M.P.E.P. § 707.07(d). Furthermore, Appellant asserts that several of the elements required by claim 20 are not taught or suggested by the prior art of record. For example, claim 20 requires, in part, "displaying a Web browser in a first window of said GUI, wherein a user browses on said Web browser to locate a Web page to be edited." Appellant points out that *XML Spy* only discloses an XML development environment. *XML Spy* at pp. 92-96. As such, the "page window editor" referred to by the Examiner is an XML editor, not a Web browser. *Id.* Therefore, a the proposed modification of *XML Spy* does not teach or suggest every element recited in claim 20. Accordingly, Appellant respectfully requests that the Board reverse this rejection.

C. Third Grounds (Claims 21-31)

Claims 21-31 are rejected under 35 U.S.C. § 103(a) as being unpatentable over *XML Spy* in view of *CuteFTP*.

1. Lack of Motivation

The rejection of claims 21-31 must be reversed because there is insufficient motivation to combine *CuteFTP* with *XML Spy*. In fact, Appellant believes that the Examiner's proposed combination of FTP client with XML editor is unnecessary, if not nonsensical. Nonetheless, the Examiner states that "[t]he motivation to combine [*CuteFTP* with *XML Spy*] is taught in CuteFTP in that it is designed to upload web compatible software to the web[,] and web compatible software is taught to be created using XML Spy." *Final Office Action* at p. 23. Appellant respectfully points out that the Examiner's proposed motivation is only a statement that the references ***can be*** combined. However, the mere fact that references ***can be*** combined does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. *See* M.P.E.P. § 2143.01. There is no suggestion or motivation, either in the prior art or in the knowledge available to a person of ordinary skill in the art, to combine an FTP client with an XML editor.

2. Lack of All Claimed Elements

Claim 21 recites, in part, "a Web browser displayed to said user in a main window of said GI . . . ." The Examiner relies solely upon *XML Spy* as meeting this element. *Final Office Action* at p. 13. As previously noted, however, neither *XML Spy* nor *CuteFTP* teach or suggest a Web browser. *XML Spy* at p. 92; *CuteFTP* at p. 12. Therefore, the combination of *XML Spy* with *CuteFTP* does not teach or suggest a Web browser. The Examiner's attempt to rebut this argument is non-responsive insofar as it does not identify any portion of *XML Spy* or *CuteFTP* that teaches or suggests a Web browser. *See Final Office Action* at p. 22.

Dependent claims 22-31 depend, either directly or indirectly, from claim 21, thus inheriting all the elements of that independent claim. As noted above, the combination of *XML Spy* with *CuteFTP* does not teach or suggest every element of independent claim 21. Thus, the combination of *XML Spy* with *CuteFTP*, even if proper, also fails to teach or suggest each and every element of dependent claims 22-31. Moreover, each of these dependent claims recite additional elements not taught or suggested by the applied art, as separately argued under the following sub-headings.

a. Dependent Claim 22

Dependent claim 22 recites "code for analyzing said Web page for page-dependent related files." The passage of *CuteFTP* relied upon by the Examiner as meeting this element discloses a combination of "scheduler and directory mirroring, [by which] users can accomplish recurring or continuous mirroring of a particular directory." *CuteFTP* at p. 9, *cited in Final Office Action* at p. 14. Appellant respectfully asserts that merely performing directory mirroring of an entire folder is very different from scanning a Web page for page dependent files. For instance, related files need not be in the same folder as the Web page itself. Meanwhile, unrelated files may be located in the same folder as the Web page. Therefore, merely mirroring a directory is not the same, or even similar, to analyzing said Web page for page-dependent related files, as recited in claim 22. The Examiner does not rely upon *XML Spy* as teaching or suggesting these elements, and Appellant asserts that it does not. Therefore, the combination of *Cute FTP* with *XML Spy*, even if proper, does not teach or suggest every element recited in claim 22.

b. Dependent Claim 23

Dependent claim 23 recites "scanning said edited source file for modified page-dependent related files prior to said automatically publishing." The passage of *CuteFTP* relied upon by the Examiner as meeting this element discloses "scripting capabilities that allow clients to automate routine tasks, such as downloading log files from a Web server or posting weekly sales reports to an FTP server." *CuteFTP* at p. 7, *cited in Final Office Action* at p. 14. Appellant has been unable to identify any passage of *CuteFTP* that teaches or suggests that its scripting feature is able to scan edited source file for modified page-dependent files, and the Examiner has not shown otherwise. Further, the Examiner does not rely upon *XML Spy* as teaching or suggesting these elements, and Appellant asserts that it does not. Therefore, the combination of *Cute FTP* with *XML Spy*, even if proper, does not teach or suggest every element recited in claim 23.

c. Dependent Claim 24

Dependent claim 24 recites "code for checking said edited source file for modified page-dependent related files prior to said automatically publishing." The Examiner states that *CuteFTP* discloses scanning a Web page for the files to be updated. *Final Office Action*

at p. 15. However, Appellant has been unable to identify any specific passage of *CuteFTP* that discloses this element, and the Examiner has not shown otherwise. Appellant respectfully asserts that *CuteFTP* only discloses mirroring a directory, and that it does not translate local links to added page-related files to reflect a location of those files on a file server. Further, the Examiner does not rely upon *XML Spy* as teaching or suggesting these elements, and Appellant asserts that it does not. Therefore, the combination of *Cute FTP* with *XML Spy*, even if proper, does not teach or suggest every element recited in claim 24.

d. Dependent Claim 26

Dependent claim 26 recites "code for translating local links to said added page-related files to reflect a location of said added page-related files on said file transfer server." The Examiner states that *CuteFTP* discloses scanning a Web page for the files to be updated. *Final Office Action* at p. 16. However, Appellant has been unable to identify any specific passage of *CuteFTP* that discloses this element, and the Examiner has not shown otherwise. Appellant respectfully asserts that *CuteFTP* only discloses mirroring a directory, and that it does not translate local links to added page-related files to reflect a location of those files on a file server, as recited in claim 26. Further, the Examiner does not rely upon *XML Spy* as teaching or suggesting these elements, and Appellant asserts that it does not. Therefore, the combination of *Cute FTP* with *XML Spy*, even if proper, does not teach or suggest every element recited in claim 26.

e. Dependent Claim 27

Dependent claim 27 recites "code for mapping addresses of said edited source file and said related files associated with said display of said Web page to an address location commensurate with said file transfer server." The Examiner states that *CuteFTP* teaches mirroring, "which makes the content of the remote drive exactly like the contents of the local drive, vice versa, or both." *Final Office Action* at p. 16. However, merely mirroring an entire folder is not the same as mapping addresses of an edited source file and related files associated with the display of a Web page to an address location commensurate with a file transfer server, as recited in claim 27. Further, the Examiner does not rely upon *XML Spy* as teaching or suggesting these elements, and Appellant asserts that it does not. Therefore, the

combination of *Cute FTP* with *XML Spy*, even if proper, does not teach or suggest every element recited in claim 27.

f. Dependent Claim 31

Dependent claim 31 recites "code for stripping said modifications to said one or more elements from said edited source responsive to said checking; and code for updating said database with said modifications." The Examiner does not properly address this claim, and in fact rejects it on the same basis as the rejection of claim 26, even though claim 31 recites entirely different elements from claim 26. Appellant respectfully asserts that *CuteFTP* does not disclose stripping modifications to one or more elements from an edited source responsive to a checking step and updating a database with the modifications, as recited in claim 31. Further, the Examiner does not rely upon *XML Spy* as teaching or suggesting these elements, and Appellant asserts that it does not. Therefore, the combination of *Cute FTP* with *XML Spy*, even if proper, does not teach or suggest every element recited in claim 31.

D. Conclusion

Appellant respectfully asserts that for at least the above reasons claims 1-31 of the present application are patentable over the rejections of record. Accordingly, Appellants respectfully request that the Board reverse these rejections and remand this application for allowance.

Dated: January 26, 2007

Respectfully submitted,

By_____
Thomas J. Meaney
Registration No.: 41,990
FULBRIGHT & JAWORSKI L.L.P.
2200 Ross Avenue, Suite 2800
Dallas, Texas 75201-2784
(214) 855-8230
(214) 855-8200 (Fax)
Attorney for Appellant

CLAIMS APPENDIX

1. (Original) A method for maintaining a Web site comprising:

browsing to a Web page to be edited;

automatically downloading a source file for said Web page from a file transfer server related to said Web page including related files associated with display of said Web page;

editing said source file; and

automatically publishing said edited source file to said file transfer server associated with said Web site including said related files associated with said display of said Web page.

2. (Original) The method of claim 1 further comprising:

scanning said Web page for page-dependent related files prior to said automatically downloading.

3. (Original) The method of claim 1 further comprising:

scanning said edited source file for modified page-dependent related files prior to said automatically publishing.

4. (Original) The method of claim 3 wherein said modified page-related files comprises one or more of:

an edited page-related file;

a deleted page-related file; and

an added page-related file.

5. (Original) The method of claim 4 further comprising:

translating local links to said added page-related files to reflect a location of said added page-related files on said file transfer server.

6. (Original) The method of claim 1 further comprising:

mapping addresses of said edited source file and said related files associated with said display of said Web page to an address location commensurate with said file transfer server.

7. (Original) The method of claim 1 wherein said Web page is generated, in part, dynamically using embedded server code and a plurality of data stored in a database.

8. (Original) The method of claim 7 further comprising:

identifying one or more elements generated from said database in said downloaded source file.

9. (Original) The method of claim 8 further comprising:

checking said edited source file for modifications to said one or more elements.

10. (Original) The method of claim 9, wherein said automatically publishing comprises

stripping said modifications to said one or more elements from said edited source responsive to said checking; and

updating said database with said modifications.

11. (Original) A computer program product having a computer readable medium with computer program logic recorded thereon for managing a Web site, said computer program product comprising:

code for browsing to a Web page to be edited;

code for automatically retrieving from a file transfer server, one or more Web source files for at least one of:

said Web page; and

one or more Web page-dependent files;

code for editing said one or more Web source files; and

code for automatically uploading said edited one or more Web source files to said file transfer server.

12. (Original) The computer program product of claim 11 further comprising:

code for inspecting said Web page for said one or more Web page-dependent files prior to said code for automatically retrieving.

13. (Original) The computer program product of claim 11 further comprising:

code for examining said edited one or more Web source files for modified Web page-dependent files prior to said code for automatically uploading.

14. (Original) The computer program product of claim 13 further comprising:

code for updating local links to one or more added Web page-dependent files included in said edited one or more Web source files to remote links reflecting file transfer server addresses of said one or more added Web page-dependent files.

15. (Original) The computer program product of claim 11 further comprising:

code for translating addresses of said edited one or more Web source files to an equivalent address for said file transfer server.

16. (Original) The computer program product of claim 11 wherein said Web page is dynamically created, in part, using embedded server code in communication with a database of information.

17. (Original) The computer program product of claim 16 further comprising:

code for identifying one or more database elements in said Web page.

18. (Original) The computer program product of claim 17 further comprising:

code for detecting said one or more database elements modified during execution of said code for editing.

19. (Original) The computer program product of claim 18, wherein said code for automatically uploading comprises:

code for extracting said modified one or more database elements from said edited one or more Web source files; and

code for updating said database with said modified one or more database elements.

20. (Original) A method for managing a Web site lifecycle from a graphical user interface (GUI) comprising:

displaying a Web browser in a first window of said GUI, wherein a user browses on said Web browser to locate a Web page to be edited;

selecting an edit indicator displayed on said GUI indicating a preference to edit said Web page;

responsive to said selecting, transitioning said first window to display a page editor;

choosing a publish indicator displayed on said GUI indicating to publish said edited Web page; and

responsive to said selecting, transitioning said first window back to display said Web browser.

21. (Original) A Web page editor comprising:

a graphical interface (GI) for receiving interaction from a user;

a Web browser displayed to said user in a main window of said GI;

an edit indicator, associated with said display of said Web browser, presented to said user on said GI, wherein said edit indicator controls execution of retrieval logic;

an edit screen replacing said Web browser in said main window responsive to said user selecting said edit indicator, wherein said user makes edits to said Web page; and

a publish indicator, associated with a display of said edit screen, presented to said user on said GI, wherein said publish indicator controls execution of said upload logic.

22. (Original) The Web page editor of claim 21 wherein said retrieval logic comprises:

code for analyzing said Web page for page-dependent related files.

23. (Original) The Web page editor of claim 22 wherein said retrieval logic comprises:

code for automatically downloading a source file for said Web page from a file transfer server related to said Web page; and

code for automatically downloading page-dependent related files.

24. (Original) The Web page editor of claim 21 wherein said upload logic comprises:

code for checking said edited source file for modified page-dependent related files prior to said automatically publishing.

25. (Original) The Web page editor of claim 24 wherein said upload logic further comprising one or more of:

an edited page-related file;

a deleted page-related file; and

an added page-related file.

26. (Original) The Web page editor of claim 25 wherein said upload logic further comprising one or more of:

code for translating local links to said added page-related files to reflect a location of said added page-related files on said file transfer server.

27. (Original) The Web page editor of claim 23 wherein said upload logic further comprising one or more of:

code for mapping addresses of said edited source file and said related files associated with said display of said Web page to an address location commensurate with said file transfer server.

28. (Original) The Web page editor of claim 23 wherein said Web page is generated, in part, dynamically using embedded server code and a plurality of data stored in a database.

29. (Original) The Web page editor of claim 28 wherein said upload logic further comprising one or more of:

code for identifying one or more elements generated from said database in said retrieved source file.

30. (Original) The Web page editor of claim 29 wherein said upload logic further comprising one or more of:

code for checking said edited source file for modifications to said one or more elements.

31. (Original) The Web page editor of claim 29 wherein said automatically uploading comprises:

code for stripping said modifications to said one or more elements from said edited source responsive to said checking; and

code for updating said database with said modifications.

EVIDENCE APPENDIX

No evidence pursuant to 37 C.F.R. §§ 1.130, 1.131, or 1.132 or entered by or relied upon by Appellee is being submitted.

RELATED PROCEEDINGS APPENDIX

No related proceedings or copies of decisions in related proceedings are being submitted.